INFUSE: A TDMA BASED DATA DISSEMINATION PROTOCOL FOR SENSOR NETWORKS

Sandeep S. Kulkarni and Mahesh Arumugam

Computer Science and Engineering
Michigan State University, East Lansing, MI 48824
Email: {sandeep,arumugam}@cse.msu.edu

ABSTRACT

Reliable dissemination of bulk data is one of the important problems in sensor networks. For example, programming or upgrading the software in sensors at run-time requires reliable dissemination of a new program across the network. In this paper, we present *Infuse*, a reliable data dissemination protocol based on TDMA based medium access layer. Although TDMA guarantees collision-freedom, unexpected channel errors (e.g., message corruption, varying signal strengths, etc) can cause random message losses. To deal with this problem, we consider two recovery schemes that use implicit acknowledgments. We also present a scheme to reduce the number of message receptions further. With this approach, sensors typically do not receive a given message multiple times. We also demonstrate that our algorithms can handle failure of sensors.

Keywords: data dissemination, network programming, time division multiple access, implicit acknowledgments, sensor networks

1. INTRODUCTION

Reliable data dissemination is one of the important problems in wireless networks, particularly in multihop sensor networks. For example, in sensor networks, reprogramming the network is often necessary since the sensors are deployed in large numbers and in hostile environments. Further, the requirements of a typical sensor network application (e.g., A Line in the Sand (LITeS) [1], habitat monitoring [2]) evolve over time and, hence, reprogramming the sensors after deployment is required. Towards this end, reliable dissemination of bulk data (i.e., the new program) is necessary.

Challenges in reliable data dissemination. One of the important challenges in data dissemination is reliable message communication. If a sensor simultaneously receives two or more messages then they collide and all the messages become incomprehensible. Also, it is often difficult to determine whether a given message was received successfully by all its intended receivers. This is due to the *hidden terminal effect*, where a given message may collide at one sensor and be correctly received at another sensor.

To provide reliable message communication, different medium access control (MAC) protocols are proposed. Collision-avoidance protocols like carrier-sense multiple access (CSMA) try to avoid collisions by sensing the medium before transmitting a message over the radio. If the medium is busy, it backs-off and tries to access the medium at a later time. However, CSMA offers only probabilistic guarantees about message communication. Most of the existing solutions for multihop data dissemination (especially for network programming) use CSMA protocol (e.g., [3]) and, hence, rely on mechanisms such as acknowledgments/negative-acknowledgments, advertisement/request schemes, and/or error correcting codes. Existing multihop reprogramming solutions based on CSMA protocol include Deluge [4], multihop over-the-air programming (MOAP) [5], and multihop network reprogramming (MNP) [6].

Collision-free MAC protocols like time division multiple access (TDMA) ensure that collisions do not occur during message communication. TDMA provides deterministic guarantees about message communication and, hence, it is desirable for reliable dissemination of bulk data in sensor networks. TDMA assigns communication slots to each sensor and ensures that simultaneous transmissions by two or more sensors will not result in collisions. Other collision-free MAC protocols include frequency division multiple access (FDMA) and code division multiple access (CDMA). With FDMA, sensors snoop different frequencies repeatedly to receive messages sent by the neighboring sensors. FDMA is often used with TDMA where each sensor knows when to listen to a particular frequency. CDMA requires special hardware for encoding and decoding messages. In order to retrieve individual messages during message communication, codes used in the network should be orthogonal to each other. Hence, CDMA is not typically desirable for resource poor sensor networks.

In this paper, we propose *Infuse*,[1] a TDMA based reliable data dissemination protocol for sensor networks. Infuse can be used with any TDMA based MAC protocol (e.g., [7, 8]). Although TDMA guarantees collision-freedom during message communication, messages can be lost due to changing link characteristics (e.g., message corruption, environmental effects on signal strengths). In order to deal with such errors, Infuse uses *implicit acknowledgments* (received by listening to the transmissions of the successors of a sensor) to recover from lost messages.

Contributions of the paper. We focus on the problem of reliable data dissemination in sensor networks. The main contributions of the paper are as follows:

- In an ideal scenario, we present a TDMA based reliable data dissemination protocol called Infuse. Infuse disseminates data in the TDMA slots assigned to each sensor. Thus, Infuse takes advantage of the reliability offered by the TDMA in providing a dissemination service. Further, we compute the analytical estimate for dissemination latency and show that simulation results closely follow analytical results.
- In presence of channel errors (e.g., message corruption, varying signal strengths, etc), random message losses occur in the network. To overcome this problem, we consider two recovery algorithms based on the sliding window protocols [9], modified to use implicit acknowledgments.

Preliminary results of this work appear as a poster in the Second ACM Conference on Embedded Networked Sensor Systems (SenSys) 2004.
This work was partially sponsored by NSF CAREER CCR-0092724, DARPA Grant OSURS01-C-1901, ONR Grant N00014-01-1-0744, NSF Equipment Grant EIA-0130724, a grant from Michigan State University.

[1] infuse v.; to cause to be permeated with something (as a principle or quality) that alters usually for the better. http://www.m-w.com/.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2004		**00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Infuse: A TDMA Based Data Dissemination Protocol for Sensor Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Michigan State University,Department of Computer Science and Engineering,East Lansing,MI,48824**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**8**	

- We show that the active radio time during dissemination is significantly lesser than the dissemination latency. We also present an optimization to reduce the active radio time further. With this approach, a sensor typically receives new capsules from only one sensor.
- We study the performance of Infuse using simulations. We verify that the data is propagated in a pipeline. Also, we show that Infuse does not have the behavior expressed in Deluge [4], where sensors along the edge of the network receive the data first before the diagonal.

Organization of the paper. In Section 2, we present the network model and state the assumptions made in this paper. In Section 3, we present the data dissemination protocol. Then, in Section 4, we discuss the properties of the protocol. Subsequently, in Section 5, we present the simulation results. In Section 6, we discuss some of the questions raised by this work and compare Infuse with other solutions for data dissemination. Finally, in Section 7, we make the concluding remarks and discuss the future work.

2. PRELIMINARIES

In this section, we present the network model and identify the assumptions made in this paper.

Base station. We assume that there exists a base station in the network that is responsible for communicating with the outside world. Also, the base station is responsible for sending commands and new tasks (for example, new program) to the network.

Network model. We assume that the sensors are deployed in a geometric topology. With help of localization service (e.g., [10]), each sensor determines its location in the field. Also, each sensor determines the location of the sensors within its neighborhood using this service. Further, each sensor classifies the sensors in its neighborhood as either their *predecessors* or their *successors*. During the dissemination process, initially, sensors listen to all the sensors in their neighborhood. Given two neighbors j and k, if k forwards a majority of packets before j then j marks k as a predecessor and k marks j as its successor. Once the predecessors and successors of j are determined, j can choose to listen to one or more of its predecessors and zero or more of its successors. Periodically, each sensor re-classifies its active neighbors as predecessors or successors. This allows a sensor to deal with the failure of some of its neighbors. Specifically, the sensor can now utilize other paths to the base station while dealing with the failure.

TDMA slot assignment. We assume that once the network is deployed, the base station assigns the slots to each sensor. TDMA slots are either pre-computed or determined dynamically by a stabilizing [11] slot assignment algorithm that ensures that correct slots are assigned in spite of transient errors such as clock drift. Examples of such protocols include [7,8]. Further, we assume that each sensor gets a fair share of the bandwidth. One way to achieve this is to ensure that between every two slots assigned to a sensor, at least one slot is assigned to its successors.

3. INFUSE: DATA DISSEMINATION PROTOCOL

In this section, we present *Infuse*, a reliable data dissemination protocol for sensor networks. First, in Section 3.1, we present Infuse in the context of an ideal environment with no channel errors. Next, in Section 3.2, we show how Infuse deals with unexpected channel errors. In Section 3.3, we discuss an algorithm for reducing the active radio time during data dissemination.

3.1. Data Dissemination in Ideal Scenario

In this section, we present the dissemination algorithm for the ideal scenario where no messages are corrupted due to varying link properties. The base station starts the dissemination process by broadcasting a *start-download* message. Whenever a sensor receives a start-download message, it prepares for data dissemination and download. The data is split into fixed size packets called *capsules*. The start-download message also includes the ID of the new data sequence and the number of capsules. The sensor then sets up the necessary pointers to the flash in order to store the capsules in their appropriate address. Finally, the sensor enqueues the start-download message in its TDMA queue.

Once the base station sends the start-download message, it sends the capsules in its subsequent TDMA slots, one capsule in each slot. Whenever a sensor receives a capsule, say, c_l, it stores c_l in the flash and it enqueues c_l in the TDMA queue. If c_l is the last capsule, it signals the application that the download is complete.

3.2. Dealing with Channel Errors

Channel errors occur in a typical sensor network due to background noise and varying properties of communication links. Although TDMA protocols proposed in [7, 8] guarantee collision-freedom, channel errors can cause random message losses. While dealing with these problems, the padding added to a message should be minimized since the payload size of a message is often limited (e.g., 29 bytes in MICA motes [12]). Further, the preamble added to a message in the lower layers of communication stack is very high (e.g., 20 bytes in MICA motes). Hence, unnecessary message communication (in terms of explicit acknowledgments) needs to be avoided.

In the dissemination algorithm presented in Section 3.1, whenever the successors of a sensor (say, j) forward the data capsule (say, c), j gets an *implicit acknowledgment* for c. We use this information to recover from lost capsules. We compare two recovery algorithms based on the sliding window protocols [9]. The recovery algorithms use implicit acknowledgments for data dissemination unlike the explicit acknowledgments required for the traditional sliding window protocols. The first algorithm, *Go-back-N* (cf. Section 3.2.1), does not add any padding to a message. The second algorithm, *selective retransmission* (cf. Section 3.2.2), adds $2b$ bits to a message, where $2b$ is the size of the window.

3.2.1. Go-Back-N Based Recovery Algorithm

In this approach, each sensor maintains a window of $2b$ capsules, where b is any integer. Specifically, each sensor maintains a window of capsules, $c_{ia}+1, \ldots, c_{ia}+2b$, where c_{ia} is the highest capsule for which j received an implicit acknowledgment from all its successors. Now, to recover from lost capsules, j will forward capsule c_f only when its successors have forwarded at least capsule $c_f - b$. Otherwise, j will start retransmitting from $c_{ia}+1$. This creates a *back pressure* in the network and, hence, the rate of dissemination of new capsules is reduced during recovery. The algorithm is shown in Figure 1.

```
sensor j:
  highest_acknowledged_capsule = min(highest capsule
      for which implicit acknowledgment is
      received from successors of j);
  next_capsule++;
  if next_capsule > highest_acknowledged_capsule + b
    next_capsule = highest_acknowledged_capsule + 1;
  enqueue next_capsule in the TDMA queue;
```

Fig. 1. Implicit acknowledgments and Go-back-N algorithm

Dealing with failed sensors. In the presence of failed sensors, neighboring sensors will not get implicit acknowledgments. To deal with this problem, whenever a sensor fails to get an implicit

acknowledgment from its successors after a fixed number of retransmissions, it declares that neighbor as failed. Now, a sensor will retransmit a capsule only when it does not receive an implicit acknowledgment from its active neighbors.

3.2.2. Selective Retransmission Based Recovery Algorithm

In this section, we present our next approach to deal with channel errors. Similar to the previous approach, each sensor maintains a window of $2b$ capsules, where b is any integer. In this approach, whenever a sensor (say, j) forwards capsule c to its successors, it piggybacks acknowledgments for capsules $c \pm x$, where $1 \leq x \leq b$. The piggybacked acknowledgments are used by the predecessors of j to determine the highest capsule for which acknowledgment is not yet received ($c_{unacked}$). To recover from lost capsules, j will forward capsule c_f only if $c_{unacked} > (c_f - b)$. Otherwise, j will retransmit $c_{unacked}$. After retransmission, j will try to forward c_f in its next TDMA slot. In other words, j selectively retransmits to recover from lost capsules. The intuition behind selective retransmission is that even if a sensor misses a capsule transmitted by one of its predecessors, it may still receive the capsule from other predecessors or its successors. (This is due to the fact the sensors may have more than one path to the base station.) Further, the piggybacked acknowledgments update the predecessors about the missing capsules at the successors. This also allows the predecessors to listen infrequently to the implicit acknowledgment of successors. Thus, selective retransmission based recovery can be used to reduce the number of message transmissions and the amount of active radio time. Further, the piggybacked acknowledgments require $2b$ bits and, hence, the padding added to a message is small. The algorithm is shown in Figure 2.

```
sensor j:
  min_unacknowledged_capsule = min(capsules for
      which implicit acknowledgments are not
      received by j from its successors);
  next_capsule = highest_capsule_forwarded + 1;
  if(next_capsule − b == min_unacknowledged capsule)
    next_capsule = min_unacknowledged_capsule;
  enqueue [next_capsule, status flag for
      next_capsule ±x, 1 ≤ x ≤ b] in the TDMA queue;
  highest_capsule_forwarded =
      max(highest_capsule_forwarded, next_capsule);
```

Fig. 2. Implicit acknowledgments and selective retransmission

Remark. In the presence of failed sensors, the modifications proposed for Go-back-N algorithm (cf. Section 3.2.1) can be applied for this approach as well.

3.3. Reducing Energy Further: Use of Preferred Predecessors

In the algorithms discussed so far, a sensor (say, j) listens in the slots assigned to its predecessors and its successors. During data dissemination, sensor j receives a given capsule from one or more of its predecessors. Also, whenever j forwards a capsule (say, c), one or more of its predecessors get implicit acknowledgment for c. To reduce these duplicates, we introduce the notion of *preferred predecessors*.

Specifically, in this approach, each sensor selects its preferred predecessor independently. For example, a sensor may choose the predecessor from which it receives the most number of new messages as its preferred predecessor. Now, whenever a sensor forwards a capsule, it includes its preferred predecessor in the message. This can achieved by $\log(q+1)$ bits, where q is the number of neighbors that a sensor has. Since the predecessors listen to the transmissions of their successors (to deal with channel errors), they learn about the sensors for whom they are the preferred predecessors. Once the preferred predecessor information is known, a sensor (say, k) will listen to the transmissions of a successor (say, j) only if j's preferred predecessor is k. Otherwise, k will not listen in the time slots assigned to j. Thus, during data dissemination, the number of message receptions is reduced by allowing only the preferred predecessors to recover lost capsules at their successors.

However, if the preferred predecessor of a sensor (say, j) failed, j cannot recover from lost capsules. To deal with this problem, other predecessors will listen to the transmissions of their successors occasionally. In other words, a sensor (say, k) will listen in the time slots assigned to j with a small probability, if k is not the preferred predecessor of j. This will allow the successors to change their preferred predecessors and recover from lost capsules and failure of preferred predecessors.

Remark. We note that the number of message communication can be reduced even further as follows. If k is the preferred predecessor of a sensor (say, j), k can choose to listen in the time slots assigned to j with a certain probability. This will allow k to listen to the transmissions of j occasionally. However, this is sufficient to recover lost capsules at j, since k learns about the lost capsules at j with the help of the implicit acknowledgments.

4. INFUSE: PROPERTIES

In this section, we discuss some of the properties of Infuse. First, we discuss how the data capsules are sent in a pipeline and estimate dissemination latency in an ideal network where no random link errors occur. Next, we argue that our approach is energy-efficient.

Pipelining. In Infuse, whenever a sensor receives a capsule, it stores the capsule in the flash at the appropriate address. Further, it forwards the capsule in the next TDMA slot. Hence, the capsules are forwarded in a pipeline fashion.

Now, we estimate the value for dissemination latency when no channel errors occur. To broadcast data over a network with c_{tot} capsules, the estimate is $((c_{tot}-1)+d)*P$ amount of time, where d is the diameter of the network and P is the period between successive TDMA slots. As a result of pipelining, the last $c_{tot}-1$ capsules can be forwarded within $(c_{tot}-1)*P$, since the base station will send one capsule per period. The first capsule takes at most $d*P$ amount of time to reach all the sensors in the network. For time slot $= 30$ ms, interference range $= 4$, data with 1000 capsules can be forwarded in a 10x10 network within 13.22 minutes.

We verify the pipelining property of Infuse using simulations in Section 5.4. Specifically, we show that data capsules are forwarded in a pipeline and this result contradicts the *dynamic behavior* presented in Deluge [4], where data capsules reach the edge of the network first before reaching the sensors in the middle.

Energy-efficiency. With TDMA, a sensor remains in active mode only in its TDMA slots (if it needs to send any capsule) and in the TDMA slots of its neighbors. Hence, in the remaining slots, sensors can save energy by turning their radio off and remaining in idle mode. If P is the period between successive TDMA slots allotted to a sensor, a sensor will have to be in active mode in its slots and in the slots of its neighbors during each period P. Thus, Infuse allows the sensors to save energy by turning their radio off in the rest of the slots. Furthermore, the optimization proposed for reducing message communication allows a sensor (say, j) to save energy by turning the radio off in the slots allotted to its successors for whom j is not the preferred predecessor. Message receptions in Infuse is within 30% excess of the analytical estimate where no channel errors occur. We compute this analytical estimate by assuming that for each capsule, a sensor (1) transmits it once, (2) receives it once from its predecessor, and (3) receives r implicit acknowledgments, where r is the number of its successors. In case

of preferred predecessors, a sensor receives $x, x \leq r$, implicit acknowledgments per capsule, where x is the number of successors for which the sensor is the preferred predecessor. Thus, Infuse disseminates data in an energy-efficient manner.

5. INFUSE: RESULTS

We simulated Infuse in prowler [13], a probabilistic wireless network simulator designed for MICA [12] based sensors. In our simulations, we use the communication model based on the TDMA algorithm proposed in [7]. We assume that the sensors can communicate with high probability among their neighbors. Furthermore, we assume that the signal from a sensor may reach sensors within distance 4 although the probability of successful communication is very low. The signal to noise ratio (SNR) of such a signal may be high enough to interfere other communication. Based on this discussion, sensors have the notion of communication range and interference range. In [14, 15], it has been shown that the ratio between interference range to effective communication range is around 3.5. In our simulations, we assume the inter-sensor separation as 10 m, communication range as 10 m, and interference range as 40 m.

Based on our experiments with MICA motes, we choose the probability of successful message communication within the communication range as 95%. In the absence of any interference, the probability of successful communication is more than 98%. Since we use TDMA for message communication, interference from other sensors does not occur. However, random channel errors can cause the reliability to go down. Hence, we choose 95% as the link reliability. This value confirms the analysis in [14, 15]. The parameters used in our simulations are listed in Table 1.

Table 1. Simulation parameters

Parameter	Value
Network parameters:	
Inter-sensor separation	10 m
Link reliability	95%
Communication range	10 m
Interference range	40 m
TDMA parameters:	
Timeslot (time to transmit one message)	30 ms
Infuse parameters:	
Capsule size	16 bytes
Maximum number of retransmissions	5
Probability of listening to successors by their non-preferred predecessors	20%

Analytical estimates. The estimate for latency, message transmissions and receptions are calculated for a specific TDMA algorithm [7]. The estimate for latency is $((c_{tot} - 1) + d) * P_b$, where c_{tot} is the number of capsules, d is the diameter of the network, and P_b is the TDMA period. For nxn grid network, $d = 2(n-1)$. The estimate for number of message transmissions by a sensor is equal to the number of capsules in the data sequence. And, the estimate for number of message receptions is 1 reception from the predecessor and 2 receptions for implicit acknowledgments from south/east neighbors per capsule.

5.1. Go-Back-N Algorithm

In this section, we present the results for Go-back-N based algorithm. Figure 3 shows the results for data dissemination with 1000 capsules. With window size$=6$, the dissemination latency is close the analytical estimate (cf. Figure 3(a)). If a sensor (say, j) missed a capsule, its predecessor (say, k) will retransmit the capsule. Since j could still get the same capsule from its other predecessors or its successors, unnecessary retransmissions are prevented with window size $= 6$. Hence, the latency is close to the analytical estimate. Since a sensor listens to only its neighbors, it turns off its radio in the slots assigned to other sensors. Thus, the active radio time of a sensor is significantly less than dissemination latency. We note that the active radio time is crucial. Specifically, the energy spent in idle listening is comparable to the energy spent in transmissions/reception. Hence, it is important that the amount of idle listening is reduced. For a 10x10 network, the active radio time is 2.32 minutes where as the latency is 15.1 minutes for disseminating data with 1000 capsules. In other words, our algorithm improves the network lifetime by 5 times when compared to the case where the radio is always on. With window size $= 20$, the latency is more than that of window size $= 6$. This is due to the fact that the recovery is too slow. Specifically, if a sensor misses a capsule, predecessors go back and start retransmitting from the missing capsule once again (although most of the retransmissions are not required).

Similar observations can be made for number of message transmissions and receptions (cf. Figures 3(b-c)).

5.2. Selective Retransmission Algorithm

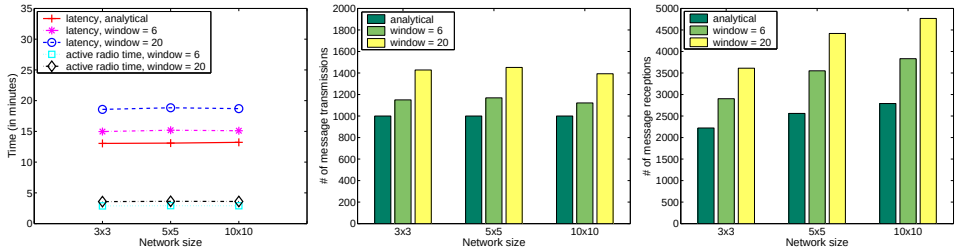
In this section, we present the results for selective retransmission based algorithm. Figure 4 shows the simulation results for data dissemination with 1000 capsules. The latency for dissemination is close to the analytical estimate for window sizes of 6 and 20 (cf. Figure 4(a)). If a sensor (say, j) misses a capsule, its predecessors selectively retransmit the lost capsule, unlike Go-back-N algorithm, thereby reducing the number of retransmissions. Thus, dissemination latency and active radio time are reduced in this recovery algorithm. For a 10x10 network, data with 1000 capsules is disseminated within 13.99 minutes. The active radio time in this case is 2.15 minutes.

And, message transmissions/receptions are within $10-25\%$ excess of analytical estimate (cf. Figures 4(b-c)).

5.3. Preferred Predecessors

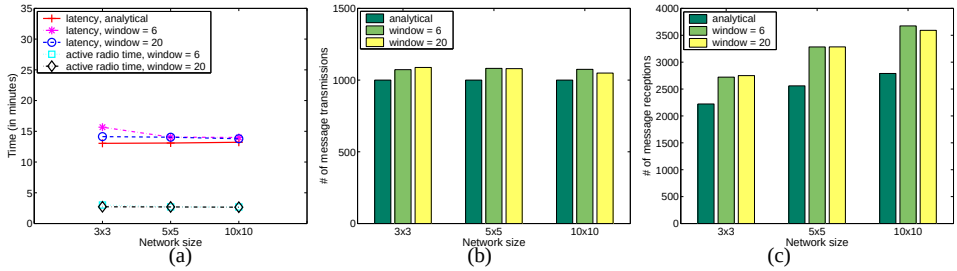
In this section, we compare the performance of Go-back-N and selective retransmission based recovery algorithms with/without preferred predecessors. Figure 5 shows the results for data dissemination with 1000 capsules. The window size used in these simulations is 6. As we can observe from the figure, the dissemination latency for selective retransmission based algorithm (SR) and selective retransmission based algorithm with preferred predecessors (SR-PP) perform better than the Go-back-N based algorithm (GBN) and Go-back-N based algorithm with preferred predecessors (GBN-PP) respectively (cf. Figure 5(a)). This is due to the fact SR and SR-PP selectively retransmits lost capsules unlike GBN and GBN-PP. Additionally, we observe that the latency for SR-PP (respectively, GBN-PP) is more than SR (respectively, GBN). This is due to the fact only the preferred predecessors are now responsible for recovering lost capsules at their successors. However, in SR and GBN, additional sources are available.

With respect to active radio time, GBN performs better than GBN-PP for large networks. Since GBN-PP retransmits starting from the lost capsule and only the preferred predecessors are responsible for recovering lost capsules, the probability of successfully retransmitting the entire window is reduced when compared to GBN. Hence, the active radio time for GBN-PP is more than that of GBN. By contrast, the active radio time for SR-PP is less than that of SR. Again, this is due to selective retransmission of lost capsules. Based on this result, we prefer SR and SR-PP compared to GBN and GBN-PP. However, GBN and GBN-PP are easy to implement and do not add any overhead to a message.



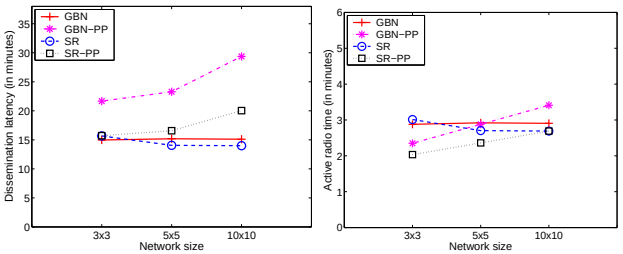
(a) (b) (c)

Fig. 3. Simulation results for disseminating data with 1000 capsules using Go-back-N algorithm. (a) dissemination latency and active radio time, (b) number of message transmissions, and (c) number of message receptions



(a) (b) (c)

Fig. 4. Simulation results for disseminating data with 1000 capsules using selective retransmission algorithm. (a) dissemination latency and active radio time, (b) number of message transmissions, and (c) number of message receptions



(a) (b)

Fig. 5. Simulation results with preferred predecessors for data with 1000 capsules. (a) dissemination latency and (b) active radio time

5.4. Pipelining and Dissemination Progress

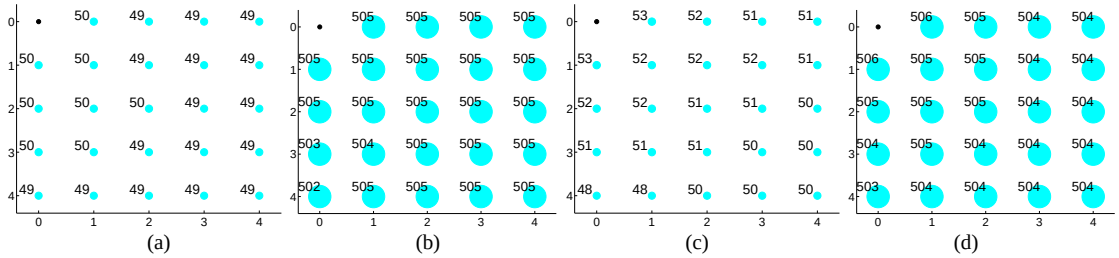
In this section, we verify the pipelining property of Infuse and also show that this result is different from the dynamic behavior discussed in Deluge [4]. Figure 6(a-b) shows the progress of data dissemination (initiated by the base station at $\langle 0, 0 \rangle$) for a data sequence consisting of 1000 capsules with Go-back-N algorithm. The window size used in these simulations is 6. At 5% of time taken to disseminate 1000 capsules, all sensors have received at least the first 49 capsules and at most 50 capsules. Similarly, at 50% of the time, all sensors have received $502 - 505$ capsules. Thus, the program capsules are transmitted in a pipeline, where each sensor acts as a source and as a receiver simultaneously.

The dissemination progress shown in Figure 6(a-b) contradicts the dynamic behavior presented in Deluge [4]. Specifically, in [4], it has been shown that the data capsules reach the edge sensors in the network first before reaching the middle of the network. This dynamic behavior causes congestion (due to CSMA based MAC) in the middle and, hence, number of message communication and latency are increased. However, with Infuse (cf. Figure 6(a-b)), we observe that all the sensors receive the data capsules at approximately the same time. And, Figure 6(c-d) shows the dissemination progress with selective retransmission based recovery algorithm. Again, this result shows that the dissemination latency along the edges is similar to the latency along the diagonal.

5.5. Effect of Window Size

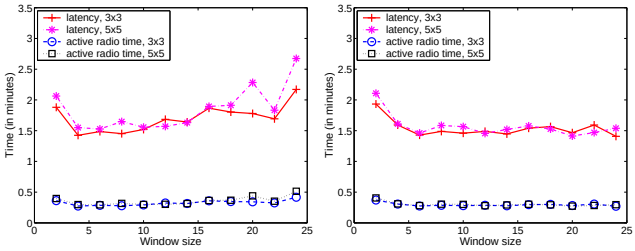
In this section, we discuss the effect of window size on the performance of the recovery algorithms. In these simulations, data consisting of 100 capsules are propagated across a 5x5 network. Figure 7(a) shows the dissemination latency and active radio time for Go-back-N based recovery algorithm. With window size $= 2$, whenever a sensor (say, k) observes that its successor (say, j) misses a capsule, it initiates recovery by retransmitting the corresponding capsule. However, j can still receive the capsule from its predecessors or its successors, as j has multiple paths to the base station. Hence, in this case, the recovery is initiated too early. Therefore, the latency (as well as the active radio time) is higher for window size $= 2$. For other values, predecessors allow the successors to recover from lost capsules through other neighbors. However, from Figure 7(a), we observe that as the window size increases, the latency also increases. This is due to the fact if j misses a capsule, its predecessor k starts retransmitting from the lost capsule, although most of the retransmissions are not necessary. Thus, with Go-back-N, we observe that the window size should be chosen such that recovery is neither initiated too early nor too late. In Figure 7(a), we note that with window size $= 4, 6, \ldots, 12$, the latency remains almost the same.

Figure 7(b) shows the effect of window size on selective retransmission based recovery algorithm. In this figure, we observe that the window size do not significantly affect the dissemination latency (and active radio time). This is due to the fact that the predecessors selectively retransmit lost capsules, unlike Go-back-N algorithm. Hence, the number of retransmissions and dissemination latency are reduced. We note that the dissemination latency for window size $= 2$ is slightly higher than that of other values. As discussed earlier in Go-back-N, in this case, if a sensor (say, j) misses a capsule, its predecessor (say, k) retransmits the corresponding capsule immediately, although j may receive the same capsule through other neighbors.



(a) (b) (c) (d)

Fig. 6. Dissemination progress for data of 1000 capsules with Go-back-N when (a) 5%, and (b) 50% of the time elapsed, and with selective retransmission when (c) 5%, and (d) 50% of the time elapsed.

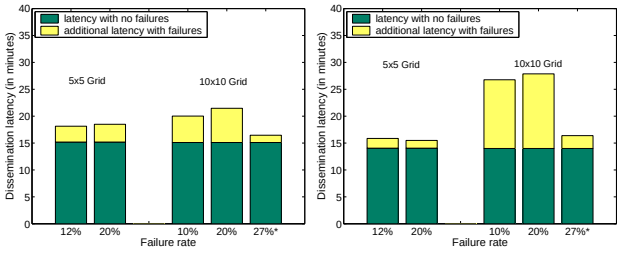


(a) (b)

Fig. 7. Effect of window size on (a) Go-back-N and (b) selective retransmission

5.6. Effect of Failed Sensors

In this section, we discuss the performance of Infuse in presence of failed sensors. In these simulations, data consisting of 1000 capsules are propagated across the network. The window size used in these simulations is 6. The failure rates used in these simulations are 10% (respectively, 12%) and 20% on a 10x10 network (respectively, 5x5 network). These values are based on the expected failure rate while deploying a sensor network (for example, in [16]). Figure 8(a) shows the effect of failed sensors on Go-back-N algorithm. We compare the dissemination latency with and without failures. In a perfect grid topology, dissemination latency for data with 1000 capsules on a 10x10 network is 15.1 minutes. In presence of 10% failed sensors, only 4.91 additional minutes are required to disseminate the data to the active sensors. Similarly, the additional delay required for 20% failure rate is around 6.36 minutes. We observe the same effect on 5x5 network.



(a) (b)

Fig. 8. Effect of failed sensors on (a) Go-back-N and (b) selective retransmission. $*$ indicates bulk failure of 3 subgrids of size 3x3.

Additionally, we did a simulation with 27% failure rate on 10x10 network. In this simulation, 3 subgrids of size 3x3 are randomly selected as the failed sensors. As we can observe from Figure 8(a), only 1.35 additional minutes are required to disseminate the data in this case. The dissemination latency in this case is significantly less than the latency in presence of 10% (and 20%) failed sensors. In case of bulk failures, the sensors need to detect the failure of the sensors along the edges only and, hence, the additional delay required in dissemination is significantly less when compared to random failures.

Figure 8(b) shows the results for selective retransmission based recovery algorithm. The additional delay required for dissemination in presence of failed sensors is more when compared to Go-back-N based recovery algorithm.

Remark. In these simulations, we assumed that the sensors fail before the dissemination starts. Even if sensors fail during the dissemination process, dissemination latency increases only by a very small percentage. Specifically, the dissemination latency is less than or equal to the latency in the case where the sensors have failed up front + the time required to detect the failure of sensors independently. The time required to detect the failure of sensors in our simulations is approximately 0.3 minutes.

5.7. Comparison: Go-Back-N and Selective Retransmission

In this section, we summarize the results of our simulations. First, in Section 5.3, to our surprise, we observe that the use of preferred predecessors does not improve the performance of the Go-back-N based recovery algorithm. This is due to the fact that duplicate sources are reduced in Go-back-N with preferred predecessors. By contrast, we do not observe this behavior with selective retransmission based recovery algorithm. Second, in Section 5.5, we observed that the window size should be chosen carefully in case of Go-back-N. Again, this parameter does not affect the performance of selective retransmission algorithm. Third, in Section 5.6, contrary to the first two observations, we note that in presence of failed sensors, with selective retransmission, the dissemination latency increases considerably. With Go-back-N, the additional time required for dissemination is in the order of $5-6$ minutes on 10x10 network. Table 2 summarizes the results.

Table 2. Comparison of recovery algorithms

	Go-back-N	Selective retransmission
Message overhead	none	2b bits, where 2b = window size
Pipelining	uniform	uniform
Preferred predecessors	not useful	reduces active radio time
Window size	affects latency; choose carefully	does not affect if window size >2
Failed sensors	tolerates random failures	increases latency considerably

6. DISCUSSION AND RELATED WORK

In this section, we discuss some of the questions raised by this work and the related work.

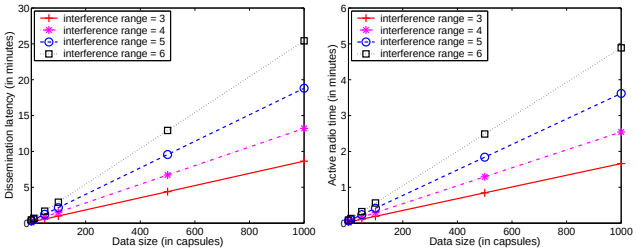
If the network has some long links, how does it affect/benefit the dissemination process?

If a sensor has a long link, it can choose to listen to the slots assigned to the corresponding sensor. Now, whenever the sensor receives new capsules from this sensor, it can save the capsule in the appropriate address and forward it, if possible, in its TDMA slot. Although the dissemination latency may be reduced by listening to such long links, the active radio time of the sensor is

increased, thereby reducing the network lifetime. Moreover, the dissemination process will not have the uniform pipelining property (as shown in Figure 6) of Infuse.

How does the protocol perform for different values of the interference range or network density?

In [14, 15], it has been shown that a signal from a sensor (say, j) reaches a sensor within distance 10 m (called, *connected region* or the communication range) with high probability. And, the sensors at distance $10 - 35$ m (called, *transitional region* or the interference range) receive the signal from j with a reduced probability. Based on this discussion, the ratio of the interference range to communication range is around 3.5. If the network density increases then this ratio increases since more number of sensors fall in the transitional region. In the simulations, we assumed this ratio to be 4. Figure 9 shows the performance of our protocol with other values. Since the simulation results are close to the analytical estimates (cf. Sections 5.1 and 5.2), we consider only the analytical estimates in this section.



(a) (b)

Fig. 9. Effect of interference range on (a) dissemination latency and (b) active radio time. Note that the scale is different.

As the ratio increases, the dissemination latency also increases. For a given interference range, the latency grows linearly with respect to the data size. Moreover, the active radio time is approximately 20% of the latency. Hence, this approach improves the network lifetime by 5 times when compared to the approach where the radio is never turned off. For example, if the ratio of interference range to communication range is 6, the latency required for disseminating data consisting of 1000 capsules is equal to 25.42 minutes. And, the active radio time per sensor is 4.89 minutes.

If the application does not use a TDMA protocol, can we use Infuse to disseminate data across the network?

Yes. Towards this end, whenever the base station requires to disseminate data, first, it allows the sensors to determine their TDMA slots (e.g., using [7, 8]). In case of SS-TDMA [7], the base station initiates a diffusing computation that assigns time slots to all sensors. This overhead is equal to the time required to disseminate one capsule across the network. And, this value depends on the interference range of the sensors. Figure 10 shows the time taken to disseminate one capsule for different interference ranges.

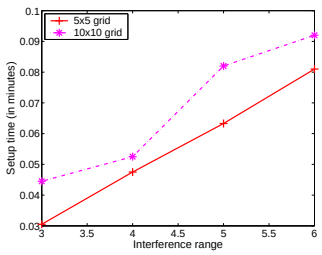


Fig. 10. TDMA setup time

From Figure 10, we observe that the time required to establish the TDMA schedule is significantly small when compared to dissemination latency for different data sizes, as shown in Figure 9. The TDMA setup time in this simulation is 0.052 minutes in a 10x10 network, where interference range $= 4$. For disseminating data with 100 capsules, this overhead is less than 3.16% of the actual dissemination latency ($= 1.66$ minutes) for Go-back-N algorithm. Hence, the overhead involved in establishing a TDMA schedule before the dissemination process does not affect the performance of our protocol.

How does the sensors determine their predecessors and successors? If the sensors cannot classify their neighbors, how does it affect the performance of Infuse?

As discussed in Section 2, given two neighbors j and k, if k forwards majority of packets before j, j marks k as its predecessor and k marks j as its successor. This allows a sensor to determine its predecessors and successors during the dissemination process. However, the sensors may not be able to classify their neighbors if the network is not uniform or randomly deployed. In this case, the sensors listen to all their neighbors during dissemination. Whenever a sensor misses a capsule, neighbors that have the corresponding capsule try to recover the capsule using the algorithms presented in Section 3.2. In this scenario, the sensors cannot take advantage of the approach proposed in Section 3.3.

What is the tradeoff in using preferred predecessors?

In Section 3.3, we proposed an optimization that allows the sensors to reduce the number of message receptions during the dissemination process. With this optimization, the effective diameter of the network may be increased, since a sensor listens only to its preferred predecessor. However, as shown in Section 4, once the first capsule is propagated across the network, the rest of the capsules can be propagated in a pipeline fashion. As observed in Figure 5(a), the dissemination latency for data containing 1000 capsules remains almost the same in 3x3, 5x5, and 10x10 networks. Hence, even if the diameter of the network increase, the dissemination latency does not increase significantly (cf. Section 5.3 for simulation results).

Can we use Infuse to disseminate data in large scale networks?

Yes. Towards this end, first, we note that the dissemination latency does not increase significantly as the network diameter increases (cf. Figure 5(a)). In applications such as [1, 16], where 1000's of sensors are deployed in a large field, the network is typically partitioned into several sections. We can use Infuse to disseminate data in each section independently and simultaneously. Thus, the whole network can be updated with the new data in the time it takes to disseminate the data across a single section.

Related work. Related work on data dissemination has been addressed for wired networks in [17] where reliable transmission of multicast messages using multiple multicast channels is proposed. One of the important concerns in data dissemination for mobile ad hoc networks is the *broadcast storm problem* [18]. Specifically, in dissemination using naive flooding based algorithms, a broadcast storm is created where redundant broadcasts, contention, and collisions occur. Infuse is not affected by the broadcast storm problem since contention/collisions are managed by TDMA.

Negotiation-based protocols that transmit advertisements or meta-data about the data sequence before initiating the actual dissemination are proposed for sensor networks (e.g., [19]). By contrast, with Infuse, negotiation using meta-data is not required.

Work related to transport protocols (e.g, [20, 21]) are also

used for bulk data dissemination. These protocols rely on negative acknowledgments for recovering from lost messages. However, Infuse does not use explicit acknowledgments/negative-acknowledgments in order to recover from lost capsules. Additionally, Infuse takes advantage of the TDMA based MAC in providing a pipelined data dissemination service.

Network programming is a special case of reliable data dissemination where code is transmitted to the network. As discussed in Introduction, unlike Infuse, most network reprogramming algorithms (e.g., [4–6]) use CSMA based MAC and, hence, rely on other mechanisms for reliable dissemination.

7. CONCLUSION AND FUTURE WORK

In this paper, we presented *Infuse*, a reliable data dissemination protocol for sensor networks. Infuse takes advantage of collision-free communication offered by TDMA. Further, Infuse recovers from random message losses caused by varying link properties and message corruption. Towards this end, we considered two recovery algorithms based on the sliding window protocols. The recovery algorithms use implicit acknowledgments to recover from lost messages unlike the explicit acknowledgments required for traditional sliding window protocols. The first algorithm, Go-back-N, recovers from random message losses with no extra information added to the payload of a message. With Go-back-N, we showed that the window size should be chosen carefully. Additionally, with the help of simulations, we observed that Go-back-N tolerates failure of sensors. The second algorithm, selective retransmissions, recovers from random message losses with $2b$ additional bits added to the payload of a message, where $2b$ is the size of the window. With selective retransmissions, we showed that window size does not significantly affect the protocol. In presence of failed sensors, to our surprise, we observed that the dissemination latency increases considerably.

In presence of no channel errors, we estimated the time required for dissemination. Also, we showed that the data capsules are transmitted in a pipeline and, hence, the latency for dissemination is reduced. We argued that Infuse disseminates data in an energy-efficient manner. Specifically, we showed that the number of message transmissions and receptions are reduced. Since Infuse uses a TDMA based MAC protocol, sensors need to listen to the radio only in the slots assigned to their neighbors. In the rest of the slots, sensors can turn off their radio. Moreover, we proposed an algorithm to reduce the number of messages further by using the notion of preferred predecessors.

We have implemented Infuse in TinyOS [22] for MICA based sensors [12]. We experimentally verified that Infuse reliably propagates a data sequence consisting of 100 capsules on 3x3 and 5x5 networks. The latency, number of message transmissions and receptions are close to the analytical results discussed in this paper.

We are currently experimenting Infuse in large scale networks (e.g., extreme scaling version of A Line in the Sand demonstration [16]). One possible scheme is to partition the network into different sections and disseminate data independently. Another interesting extension is to support dynamic adaptation in sensor network applications.

8. REFERENCES

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, "A line in the sand: A wireless sensor network for target detection, classification, and tracking," *Computer Networks (Elsevier)*, vol. 46, no. 5, pp. 605–634, December 2004.

[2] A. Mairwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," *In Proceedings of the Workshop On Wireless Sensor Networks and Applications*, 2002.

[3] A. Woo and D. Culler, "A transmission control scheme for media access in sensor networks," *In Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, 2001.

[4] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale," *In Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, 2004.

[5] T. Stathopoulos, J. Heidemann, and D. Estrin, "A remote code update mechanism for wireless sensor networks," Tech. Rep. CENS-TR-30, Center for Embedded Networked Computing, 2003.

[6] S. S. Kulkarni and L. Wang, "MNP: Multihop network reprogramming service for sensor networks," Tech. Rep. MSU-CSE-04-19, Michigan State University, May 2004.

[7] S. S. Kulkarni and M. Arumugam, "SS-TDMA: A self-stabilizing MAC for sensor networks," in *Sensor Network Operations*. IEEE Press, 2004, to appear.

[8] T. Herman and S. Tixeuil, "A distributed TDMA slot assignment algorithm for wireless sensor networks," *In Proceedings of the Workshop on Algorithmic Aspects of Wireless Sensor Networks*, 2004.

[9] A. S. Tanenbaum, *Computer Networks*, Prentice Hall, 4th edition, March 2003.

[10] G. Agha, W. Kim, Y. Kwon, K. Mechitov, and S. Sundresh, "Evaluation of localization services," 2003, DARPA NEST Program.

[11] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, 1974.

[12] J. Hill and D. E. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[13] G. Simon, P. Volgyesi, M. Maroti, and A. Ledeczi, "Simulation-based optimization of communication protocols for large-scale wireless sensors networks," *IEEE Aerospace Conference*, March 2003.

[14] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "An empirical study of epidemic algorithms in large scale multihop wireless networks," Tech. Rep. IRB-TR-02-003, Intel Research, March 2002.

[15] M. Zuniga and B. Krishnamachari, "Analyzing the transitional region in low power wireless links," *In Proceedings of the IEEE Conference on Sensor and Ad hoc Communications and Networks*, 2004.

[16] The Ohio State University NEST Team, "ExScal: Extreme scaling in sensor network for target detection, classification, tracking," 2004, ongoing project, DARPA, `http://nest.netlab.ohio-state.edu`.

[17] S. K. Kasera, G. Hjálmtýsson, D. F. Towsley, and J. F. Kurose, "Scalable reliable multicast using multiple multicast channels," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, 2000.

[18] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Wireless Networks*, vol. 8, no. 2-3, pp. 153–167, 2002.

[19] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 169–185, 2002.

[20] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A reliable transport protocol for wireless sensor networks," *Workshop on Wireless Sensor Networks and Applications*, 2002.

[21] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," *In Proceedings of the First International Workshop on Sensor Net Protocols and Applications*, April 2003.

[22] "TinyOS: A component-based OS for the networked sensor regime," `http://www.tinyos.net`.